EXTENDING COMPACT-TABLE TO BASIC SMART TABLES

CP2017

Hélène Verhaeghe[1], Christophe Lecoutre[2], Yves Deville[1], Pierre Schaus[3]

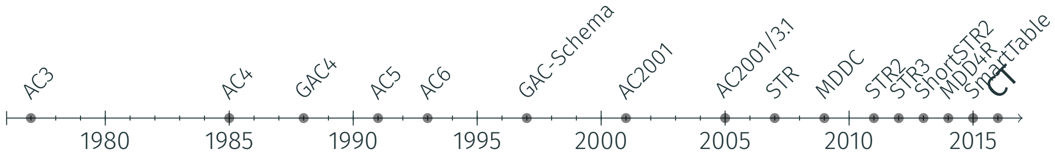
1 September 2017

[1]UCLouvain, ICTEAM, Place Sainte Barbe 2, 1348 Louvain-la-Neuve, Belgium, $\{firstname.lastname\}@uclouvain.be$
[2]CRIL-CNRS UMR 8188, Université d'Artois, F-62307 Lens, France, $lecoutre@cril.fr$

THE TABLE CONSTRAINT

Tables are the oldest most used CP constraints

$$(x, y, z) \in \begin{array}{c|ccc}
\hline
 & x & y & z \\
\hline
\tau_1 & a & a & a \\
\tau_2 & d & d & a \\
\tau_3 & c & e & b \\
\vdots & \vdots & \vdots & \vdots \\
\hline
\end{array}$$



2016 : New algorithm ! CompactTable [CP2016], based on bitwise operations, completely outperformed existing algorithms.

COMPACT-TABLE [CP2016]

TABLES : DEFINITION AND SET VISUALISATION

A Table

contains Tuples

	x	y	z
τ_1	a	a	a
τ_2	c	e	b
τ_3	d	d	a
$\vdots$	$\vdots$	$\vdots$	$\vdots$

$$\begin{tikzcd}
\tau_2 \arrow[rr] & & (c,\, e,\, b) \arrow[dl] \arrow[d] \arrow[dr] & \\
& (x,\, c) & (y,\, e) & (z,\, b)
\end{tikzcd}$$

TABLES : DEFINITION AND SET VISUALISATION

A Table

	x	y	z
τ_1	a	a	a
τ_2	c	e	b
τ_3	d	d	a
$\vdots$	$\vdots$	$\vdots$	$\vdots$

contains Tuples

$$\begin{tikzcd}
\tau_2 \arrow[rr] & & (c,\ e,\ b) \arrow[dl] \arrow[d] \arrow[dr] & \\
& (x,\ c) & (y,\ e) & (z,\ b)
\end{tikzcd}$$

Set of the tuples of the Table

(x,a)	(x,d)
(x,b)	(x,e)
(x,c)	(x,f)

TABLES : DEFINITION AND SET VISUALISATION

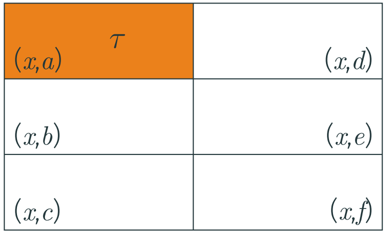
A Table

	x	y	z
τ_1	a	a	a
τ_2	c	e	b
τ_3	d	d	a
$\vdots$	$\vdots$	$\vdots$	$\vdots$

contains Tuples

$$\begin{tikzcd} & (c,\ e,\ b) \arrow[dl] \arrow[d] \arrow[dr] & \\ (x,\ c) & (y,\ e) & (z,\ b) \end{tikzcd}$$

Set of the tuples of the Table



For example: $\tau = (a,\ b,\ c)$

TABLES : DEFINITION AND SET VISUALISATION

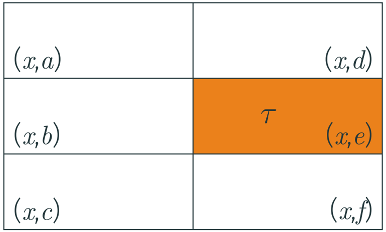
A Table

	x	y	z
τ_1	a	a	a
τ_2	c	e	b
τ_3	d	d	a
$\vdots$	$\vdots$	$\vdots$	$\vdots$

contains Tuples

$$\begin{tikzcd}
 & (c,\ e,\ b) \arrow[dl] \arrow[d] \arrow[dr] & \\
(x,\ c) & (y,\ e) & (z,\ b)
\end{tikzcd}$$

Set of the tuples of the Table



For example: $\tau = (e,\ c,\ a)$

COMPACT-TABLE

1. Which tuples are still valid?

2. Which values are no more supported?

COMPACT-TABLE

1. Which tuples are still valid?

   Update phase

2. Which values are no more supported?

COMPACT-TABLE

1. Which tuples are still valid?

   Update phase

2. Which values are no more supported?

   Propagation phase

UPDATE : INTUITION

Goal of Update

Knowing which tuples are still valid

	Dom	Δ
x	$\{\ a,\ b,\ c\ \}$	$\{\ \}$
y	$\{\ a,\ b,\ c\ \}$	$\{\ \}$
z	$\{\ a,\ b,\ c\ \}$	$\{\ \}$

State

	x	y	z	
τ_1	a	a	a	✓
τ_2	a	b	c	✓
τ_3	c	a	b	✓
τ_4	b	c	c	✓
τ_5	a	c	a	✓

Table

τ_1	τ_2	τ_3	τ_4	τ_5
1	1	1	1	1

`currTable`

UPDATE : INTUITION

Goal of Update

Knowing which tuples are still valid

	Dom	Δ
x	$\{ a, \cancel{b}, c \}$	$\{ b \}$
y	$\{ a, b, c \}$	$\{ \}$
z	$\{ a, b, c \}$	$\{ \}$

State

	x	y	z	
τ_1	a	a	a	✓
τ_2	a	b	c	✓
τ_3	c	a	b	✓
τ_4	b	c	c	✗
τ_5	a	c	a	✓

Table

τ_1	τ_2	τ_3	τ_4	τ_5
1	1	1	0	1

`currTable`

UPDATE : INTUITION

Goal of Update

Knowing which tuples are still valid

	Dom	Δ
x	$\{\ a,\ c\ \}$	$\{\ b\ \}$
y	$\{\ a,\ b,\ c\ \}$	$\{\ \}$
z	$\{\ a,\ b,\ \cancel{c}\ \}$	$\{\ c\ \}$

State

	x	y	z	
τ_1	a	a	a	✓
τ_2	a	b	c	✗
τ_3	c	a	b	✓
τ_4	b	c	c	
τ_5	a	c	a	✓

Table

τ_1	τ_2	τ_3	τ_4	τ_5
1	0	1	0	1

`currTable`

UPDATE : INTUITION

Goal of Update

Knowing which tuples are still valid

	Dom	Δ
x	$\{$ ~~a~~, c $\}$	$\{ a, b \}$
y	$\{ a, b, c \}$	$\{ \}$
z	$\{ a, b \}$	$\{ c \}$

State

	x	y	z	
τ_1	a	a	a	✗
τ_2	a	b	c	
τ_3	c	a	b	✓
τ_4	b	c	c	
τ_5	a	c	a	✗

Table

τ_1	τ_2	τ_3	τ_4	τ_5
0	0	1	0	0

`currTable`

SUPPORTS

	x	y	z
τ_1	a	a	a
τ_2	a	b	c
τ_3	c	a	b
τ_4	b	c	c
τ_5	a	c	a

Table

	τ_1	τ_2	τ_3	τ_4	τ_5
(x,a)					
(x,b)					
(x,c)					
(y,a)					
(y,b)					
(y,c)					
(z,a)					
(z,b)					
(z,c)					

`support`

SUPPORTS

	x	y	z
τ_1	a	a	a
τ_2	a	b	c
τ_3	c	a	b
τ_4	b	c	c
τ_5	a	c	a

Table

	τ_1	τ_2	τ_3	τ_4	τ_5
(x,a)	1				
(x,b)	0				
(x,c)	0				
(y,a)	1				
(y,b)	0				
(y,c)	0				
(z,a)	1				
(z,b)	0				
(z,c)	0				

`support`

SUPPORTS

	x	y	z
τ_1	a	a	a
τ_2	a	b	c
τ_3	c	a	b
τ_4	b	c	c
τ_5	a	c	a

Table

	τ_1	τ_2	τ_3	τ_4	τ_5
(x,a)	1	1			
(x,b)	0	0			
(x,c)	0	0			
(y,a)	1	0			
(y,b)	0	1			
(y,c)	0	0			
(z,a)	1	0			
(z,b)	0	0			
(z,c)	0	1			

`support`

SUPPORTS

	x	y	z
τ_1	a	a	a
τ_2	a	b	c
τ_3	c	a	b
τ_4	b	c	c
τ_5	a	c	a

Table

	τ_1	τ_2	τ_3	τ_4	τ_5
(x,a)	1	1	0		
(x,b)	0	0	0		
(x,c)	0	0	1		
(y,a)	1	0	1		
(y,b)	0	1	0		
(y,c)	0	0	0		
(z,a)	1	0	0		
(z,b)	0	0	1		
(z,c)	0	1	0		

`support`

SUPPORTS

	x	y	z
τ_1	a	a	a
τ_2	a	b	c
τ_3	c	a	b
τ_4	b	c	c
τ_5	a	c	a

Table

	τ_1	τ_2	τ_3	τ_4	τ_5
(x,a)	1	1	0	0	
(x,b)	0	0	0	1	
(x,c)	0	0	1	0	
(y,a)	1	0	1	0	
(y,b)	0	1	0	0	
(y,c)	0	0	0	1	
(z,a)	1	0	0	0	
(z,b)	0	0	1	0	
(z,c)	0	1	0	1	

`support`

SUPPORTS

	x	y	z
τ_1	a	a	a
τ_2	a	b	c
τ_3	c	a	b
τ_4	b	c	c
τ_5	a	c	a

Table

	τ_1	τ_2	τ_3	τ_4	τ_5
(x,a)	1	1	0	0	1
(x,b)	0	0	0	1	0
(x,c)	0	0	1	0	0
(y,a)	1	0	1	0	0
(y,b)	0	1	0	0	0
(y,c)	0	0	0	1	1
(z,a)	1	0	0	0	1
(z,b)	0	0	1	0	0
(z,c)	0	1	0	1	0

`support`

SUPPORTS

	x	y	z
τ_1	a	a	a
τ_2	a	b	c
τ_3	c	a	b
τ_4	b	c	c
τ_5	a	c	a

Table

	τ_1	τ_2	τ_3	τ_4	τ_5
(x,a)	1	1	0	0	1
(x,b)	0	0	0	1	0
(x,c)	0	0	1	0	0
(y,a)	1	0	1	0	0
(y,b)	0	1	0	0	0
(y,c)	0	0	0	1	1
(z,a)	1	0	0	0	1
(z,b)	0	0	1	0	0
(z,c)	0	1	0	1	0

`support`

Set of Tuples
(x,a)
(x,b)
(x,c)

Sets

SUPPORTS

	x	y	z
τ_1	a	a	a
τ_2	a	b	c
τ_3	c	a	b
τ_4	b	c	c
τ_5	a	c	a

Table

	τ_1	τ_2	τ_3	τ_4	τ_5
(x,a)	1	1	0	0	1
(x,b)	0	0	0	1	0
(x,c)	0	0	1	0	0
(y,a)	1	0	1	0	0
(y,b)	0	1	0	0	0
(y,c)	0	0	0	1	1
(z,a)	1	0	0	0	1
(z,b)	0	0	1	0	0
(z,c)	0	1	0	1	0

`support`

Set of Tuples

(x,a)	τ_1	τ_2	τ_5
(x,b)			
(x,c)			

Sets

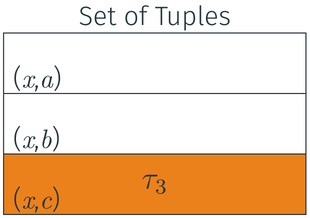
SUPPORTS

	x	y	z
τ_1	a	a	a
τ_2	a	b	c
τ_3	c	a	b
τ_4	b	c	c
τ_5	a	c	a

Table

	τ_1	τ_2	τ_3	τ_4	τ_5
(x,a)	1	1	0	0	1
(x,b)	0	0	0	1	0
(x,c)	0	0	1	0	0
(y,a)	1	0	1	0	0
(y,b)	0	1	0	0	0
(y,c)	0	0	0	1	1
(z,a)	1	0	0	0	1
(z,b)	0	0	1	0	0
(z,c)	0	1	0	1	0

`support`



Sets

CT : CLASSICAL UPDATE

Set of Tuples in Table

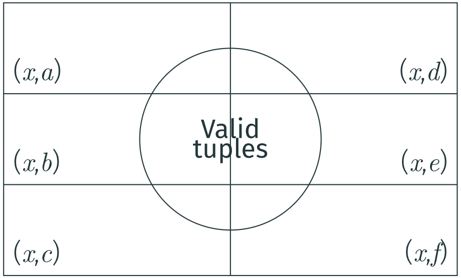
(x,a)	(x,d)
(x,b)	(x,e)
(x,c)	(x,f)

Goal of the update

Remove invalid tuples from `currTable`

CT : CLASSICAL UPDATE

Set of Tuples in Table

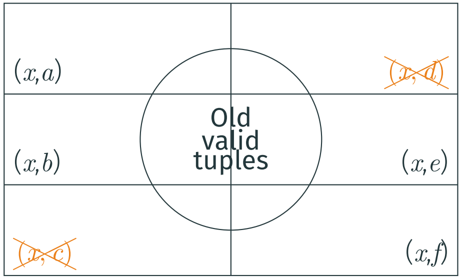


Goal of the update

Remove invalid tuples from `currTable`

CT : CLASSICAL UPDATE

Set of Tuples in Table

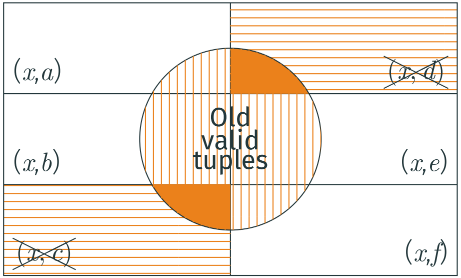


Goal of the update

Remove invalid tuples from `currTable`

CT : CLASSICAL UPDATE

Set of Tuples in Table

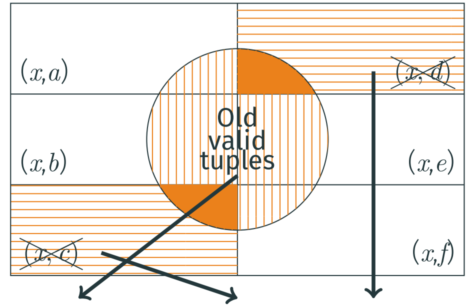


Goal of the update

Remove invalid tuples from `currTable`

CT : CLASSICAL UPDATE

Set of Tuples in Table



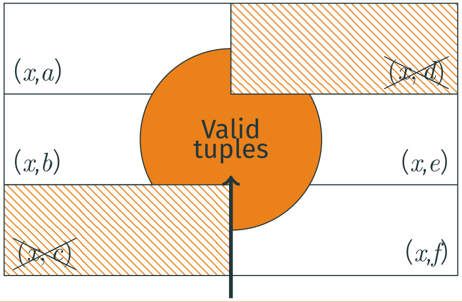
`currTable` $\cap$ (`support`$[x, c] \cup$ `support`$[x, d])^C$

Goal of the update

Remove invalid tuples from `currTable`

CT : CLASSICAL UPDATE

Set of Tuples in Table

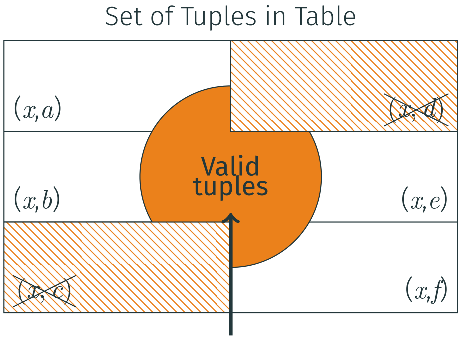


$\boxed{\texttt{currTable} \cap (\texttt{support}[x, c] \cup \texttt{support}[x, d])^C}$

Goal of the update

Remove invalid tuples from `currTable`

CT : CLASSICAL UPDATE



$\boxed{\texttt{currTable} \cap (\texttt{support}[x,c] \cup \texttt{support}[x,d])^C}$

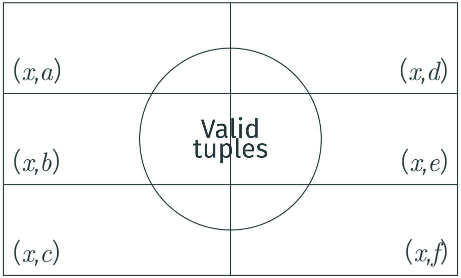
Goal of the update

Remove invalid tuples from `currTable`

```
Algorithm: ClassicalUpdate(x)
1 mask ← $0$ ;
2 foreach value $a \in \Delta_x$ do
3     mask ← mask | supports$[x, a]$ ;
4 mask ← $\sim$ mask ;
5 currTable ← currTable & mask ;
```

CT : RESET UPDATE

Set of Tuples in Table

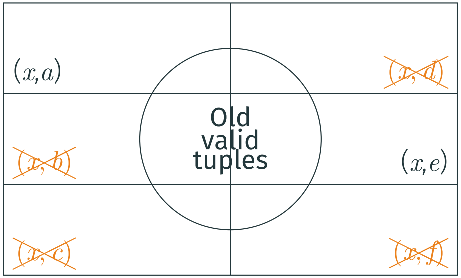


Goal of the update

Remove invalid tuples from `currTable`

CT : RESET UPDATE

Set of Tuples in Table

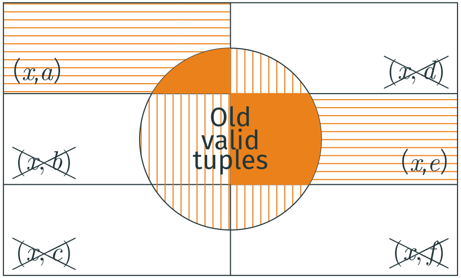


Goal of the update

Remove invalid tuples from `currTable`

CT : RESET UPDATE

Set of Tuples in Table

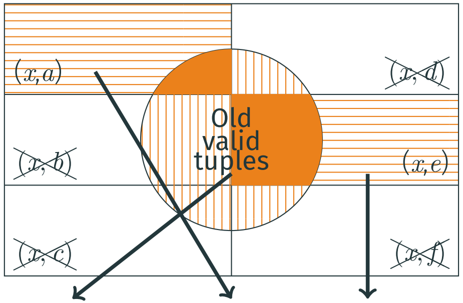


Goal of the update

Remove invalid tuples from `currTable`

CT : RESET UPDATE

Set of Tuples in Table



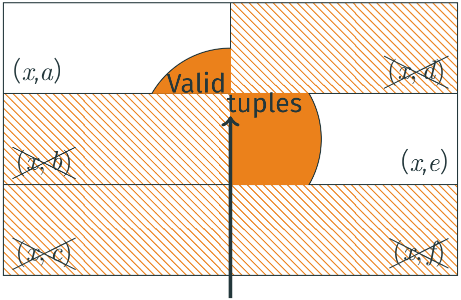
$\texttt{currTable} \cap (\texttt{support}[x,a] \cup \texttt{support}[x,e])$

Goal of the update

Remove invalid tuples from `currTable`

CT : RESET UPDATE

Set of Tuples in Table



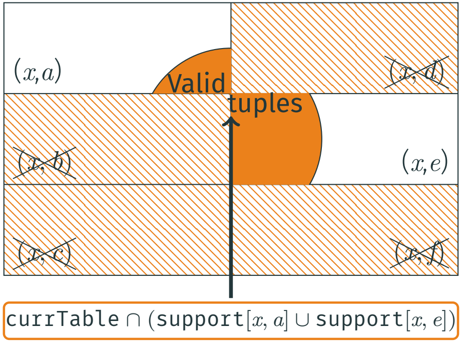
$\boxed{\texttt{currTable} \cap (\texttt{support}[x, a] \cup \texttt{support}[x, e])}$

Goal of the update

Remove invalid tuples from `currTable`

CT : RESET UPDATE

Set of Tuples in Table



(x,a)
(x,d)
(x,b)
(x,e)
(x,c)
(x,f)
Valid tuples
$\texttt{currTable} \cap (\texttt{support}[x,a] \cup \texttt{support}[x,e])$

Goal of the update

Remove invalid tuples from `currTable`

```
Algorithm: ResetUpdate(x)
1 mask ← $0$ ;
2 foreach value $a \in dom(x)$ do
3     mask ← mask | supports$[x,a]$ ;
4 currTable ← currTable & mask ;
```

CT: WHICH UPDATE TO CHOOSE?

- Classical update :

$$\mathcal{O}(|\Delta_x|)$$

- Reset update :

$$\mathcal{O}(|dom(x)|)$$

Goal of the update

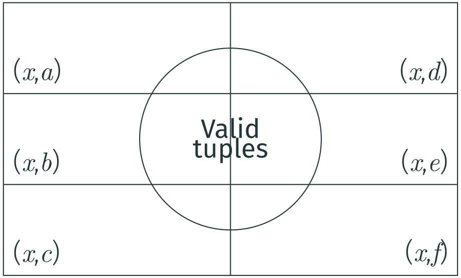
Remove invalid tuples from `currTable`

Algorithm: Update(x)

```
foreach variable $x \in$ scp where $|\Delta_x| > 0$ do
    if $|\Delta_x| < |dom(x)|$ then
        ClassicalUpdate(x);
    else
        ResetUpdate(x);
```

CT : PROPAGATION

Set of Tuples in Table

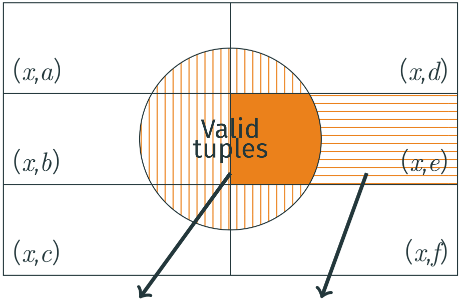


Goal of the propagation

Remove unsupported values

CT : PROPAGATION

Set of Tuples in Table



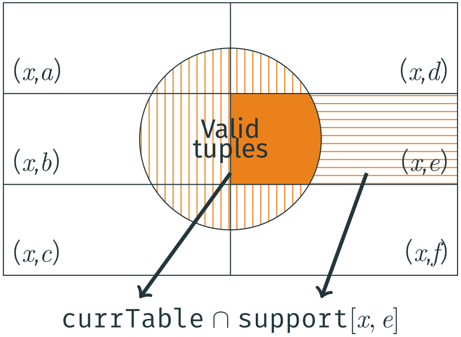
`currTable` $\cap$ `support`$[x, e]$

Goal of the propagation

Remove unsupported values

CT : PROPAGATION

Set of Tuples in Table



$\texttt{currTable} \cap \texttt{support}[x, e]$

Goal of the propagation

Remove unsupported values

Algorithm: Propagate()

```
1 foreach variable $x \in$ scp do
2     foreach value $a \in dom(x)$ do
3         if currTable & supports$[x, a] = 0$
            then
4             $dom(x) \leftarrow dom(x) \setminus \{a\}$ ;
```

COMPACT-TABLE FOR BASIC SMART TUPLES

BASIC SMART TUPLES

A Basic Smart Table

	x	y	z
τ_1	$*$	$*$	$\in \{a, b\}$
τ_2	$\neq a$	c	$\leq a$
τ_3	b	$*$	$*$
τ_4	$\geq c$	$\neq b$	a
	$\vdots$	$\vdots$	$\vdots$

BASIC SMART TUPLES

A Basic Smart Table contains Smart Elements representing multiples values

	x	y	z
τ_1	$*$	$*$	$\in \{a, b\}$
τ_2	$\neq a$	c	$\leq a$
τ_3	b	$*$	$*$
τ_4	$\geq c$	$\neq b$	a
	$\vdots$	$\vdots$	$\vdots$

BASIC SMART TUPLES

A Basic Smart Table

	x	y	z
τ_1	$*$	$*$	$\in \{a, b\}$
τ_2	$\neq a$	c	$\leq a$
τ_3	b	$*$	$*$
τ_4	$\geq c$	$\neq b$	a
	$\vdots$	$\vdots$	$\vdots$

contains Smart Elements

single value: e

representing multiples values

$\cancel{a}, \cancel{b}, \cancel{c}, \cancel{d}, e, \cancel{f}$

BASIC SMART TUPLES

A Basic Smart Table

	x	y	z
τ_1	$*$	$*$	$\in \{a, b\}$
τ_2	$\neq a$	c	$\leq a$
τ_3	b	$*$	$*$
τ_4	$\geq c$	$\neq b$	a
	$\vdots$	$\vdots$	$\vdots$

contains Smart Elements

single value: e

universal value: $*$

representing multiples values

$\cancel{a}, \cancel{b}, \cancel{c}, \cancel{d}, e, \cancel{f}$

a, b, c, d, e, f

BASIC SMART TUPLES

A Basic Smart Table

	x	y	z
τ_1	$*$	$*$	$\in \{a, b\}$
τ_2	$\neq a$	c	$\leq a$
τ_3	b	$*$	$*$
τ_4	$\geq c$	$\neq b$	a
	$\vdots$	$\vdots$	$\vdots$

contains Smart Elements

single value: e	$\cancel{a}, \cancel{b}, \cancel{c}, \cancel{d}, e, \cancel{f}$
universal value: $*$	a, b, c, d, e, f
exclusion: $\neq e$	$a, b, c, d, \cancel{e}, f$

representing multiples values

BASIC SMART TUPLES

A Basic Smart Table contains Smart Elements representing multiples values

	x	y	z
τ_1	$*$	$*$	$\in \{a, b\}$
τ_2	$\neq a$	c	$\leq a$
τ_3	b	$*$	$*$
τ_4	$\geq c$	$\neq b$	a
	$\vdots$	$\vdots$	$\vdots$

single value: e	$\cancel{a}, \cancel{b}, \cancel{c}, \cancel{d}, e, \cancel{f}$
universal value: $*$	a, b, c, d, e, f
exclusion: $\neq e$	$a, b, c, d, \cancel{e}, f$
upper bound: $\leq c$	$a, b, c, \cancel{d}, \cancel{e}, \cancel{f}$

BASIC SMART TUPLES

A Basic Smart Table contains Smart Elements representing multiples values

	x	y	z
τ_1	$*$	$*$	$\in \{a, b\}$
τ_2	$\neq a$	c	$\leq a$
τ_3	b	$*$	$*$
τ_4	$\geq c$	$\neq b$	a
	$\vdots$	$\vdots$	$\vdots$

single value: e	$\cancel{a}, \cancel{b}, \cancel{c}, \cancel{d}, e, \cancel{f}$
universal value: $*$	a, b, c, d, e, f
exclusion: $\neq e$	$a, b, c, d, \cancel{e}, f$
upper bound: $\leq c$	$a, b, c, \cancel{d}, \cancel{e}, \cancel{f}$
lower bound: $\geq c$	$\cancel{a}, \cancel{b}, c, d, e, f$

BASIC SMART TUPLES

A Basic Smart Table contains Smart Elements representing multiples values

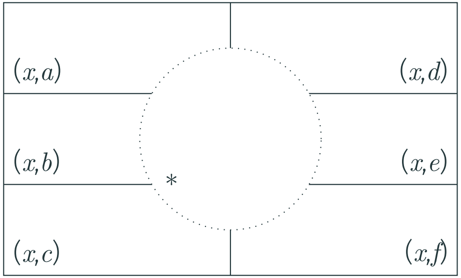
	x	y	z
τ_1	$*$	$*$	$\in \{a, b\}$
τ_2	$\neq a$	c	$\leq a$
τ_3	b	$*$	$*$
τ_4	$\geq c$	$\neq b$	a
	$\vdots$	$\vdots$	$\vdots$

single value: e	$\cancel{a}, \cancel{b}, \cancel{c}, \cancel{d}, e, \cancel{f}$
universal value: $*$	a, b, c, d, e, f
exclusion: $\neq e$	$a, b, c, d, \cancel{e}, f$
upper bound: $\leq c$	$a, b, c, \cancel{d}, \cancel{e}, \cancel{f}$
lower bound: $\geq c$	$\cancel{a}, \cancel{b}, c, d, e, f$
set: $\in \{a, c, d\}$	$a, \cancel{b}, c, d, \cancel{e}, \cancel{f}$

CT^{bs}: $*$ [AAAI17]

classical update

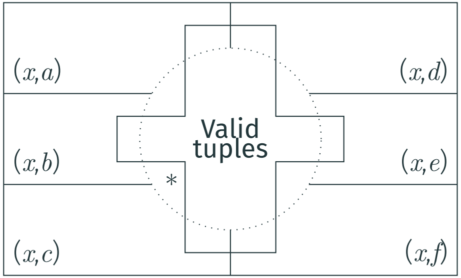
Set of Tuples in Table



CT^{bs}: $*$ [AAAI17]

classical update

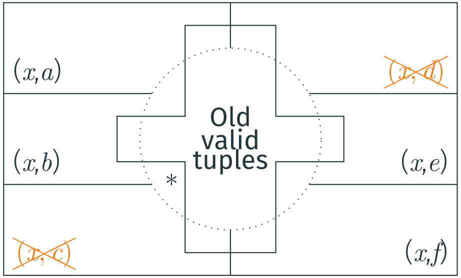
Set of Tuples in Table



CT^{bs}: $*$ [AAAI17]

classical update

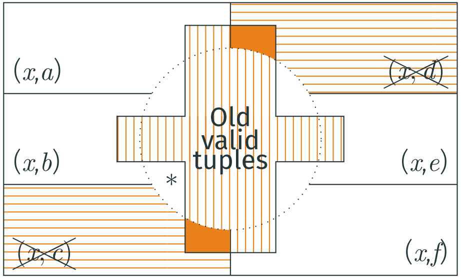
Set of Tuples in Table



CT^{bs}: $*$ [AAAI17]

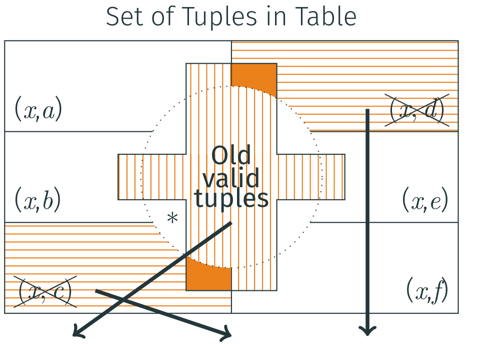
classical update

Set of Tuples in Table



CT^{bs}: $*$ [AAAI17]

classical update

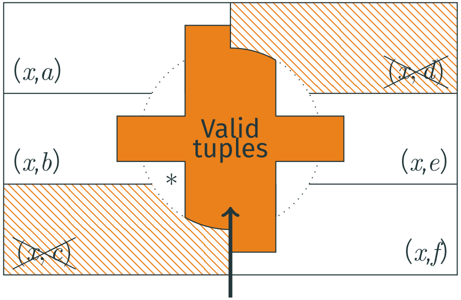


`currTable` $\cap$ (`support`$^*[x,c] \cup$ `support`$^*[x,d])^C$

CT^{bs}: $*$ [AAAI17]

classical update

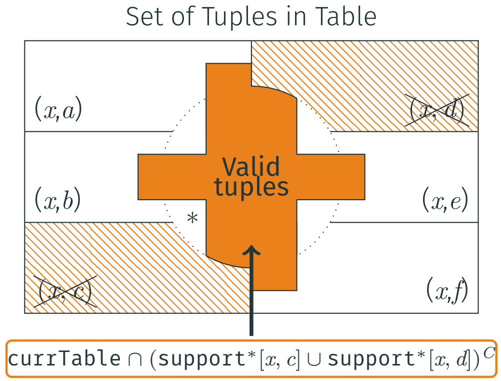
Set of Tuples in Table



$\texttt{currTable} \cap (\texttt{support}^*[x,c] \cup \texttt{support}^*[x,d])^C$

CT^{bs}: $*$ [AAAI17]

classical update


Set of Tuples in Table
(x,a)
(x,b)
(x,c)
(x,d)
(x,e)
(x,f)
Valid tuples
$*$
$\texttt{currTable} \cap (\texttt{support}^*[x,c] \cup \texttt{support}^*[x,d])^C$

```
Algorithm: ClassicalUpdate(x)
1 mask ← $0$ ;
2 foreach value $a \in \Delta_x$ do
3     mask ← mask | supports$^*[x,a]$ ;
4 mask ← ∼ mask ;
5 currTable ← currTable & mask ;
```

CT^{bs}: $\neq$

classical update

$|dom(x)| == 0$

$|dom(x)| > 1$

$|dom(x)| == 1$

CT^{bs}: $\neq$

classical update

$|dom(x)| == 0$

Trivial!
Handled by variable x

$|dom(x)| == 1$

$|dom(x)| > 1$

CT^{bs}: $\neq$

classical update

$$|dom(x)| == 0$$

Trivial!
Handled by variable x

$$|dom(x)| == 1$$

$|\Delta_x| \geq |dom(x)|$ always true!
`ResetUpdate(x)` used
and already working!

$$|dom(x)| > 1$$

CT^{bs}: $\neq$

classical update

$|dom(x)| == 0$

Trivial!
Handled by variable x

$|dom(x)| == 1$

$|\Delta_x| \geq |dom(x)|$ always true!
`ResetUpdate(x)` used
and already working!

$|dom(x)| > 1$

If $|\Delta_x| < |dom(x)|$

Tuple always valid!

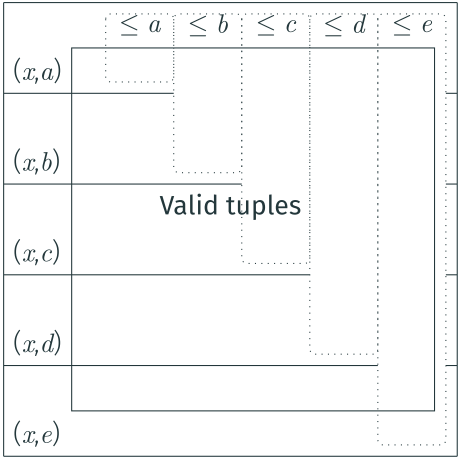
At least one valid value
`support`$^*[x][\tau] = 0$

If $|\Delta_x| \geq |dom(x)|$

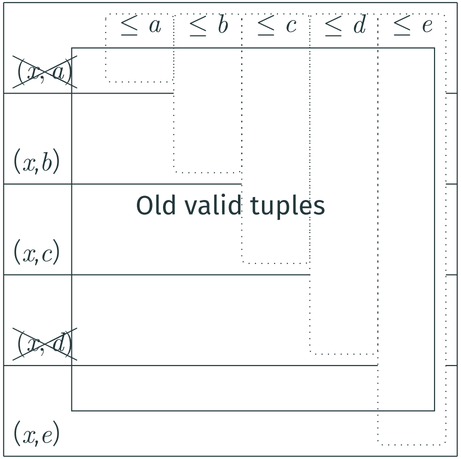
`ResetUpdate(x)` used
and already working!

CT^{bs}: $\leq$ & $\geq$

classical update

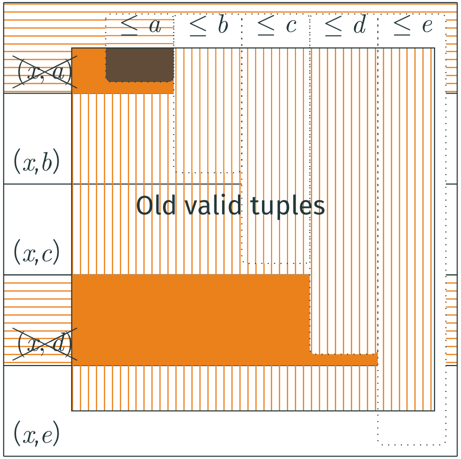


CT^{bs}: $\leq$ & $\geq$


$\leq a$
$\leq b$
$\leq c$
$\leq d$
$\leq e$
$\cancel{(x,a)}$
(x,b)
(x,c)
$\cancel{(x,d)}$
(x,e)
Old valid tuples

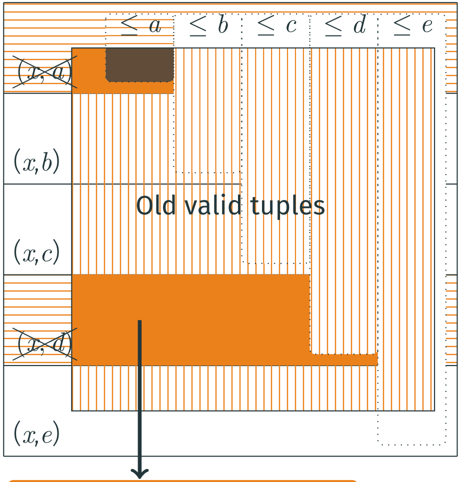
CT^{bs}: $\leq$ & $\geq$

classical update



CT^{bs}: $\leq$ & $\geq$

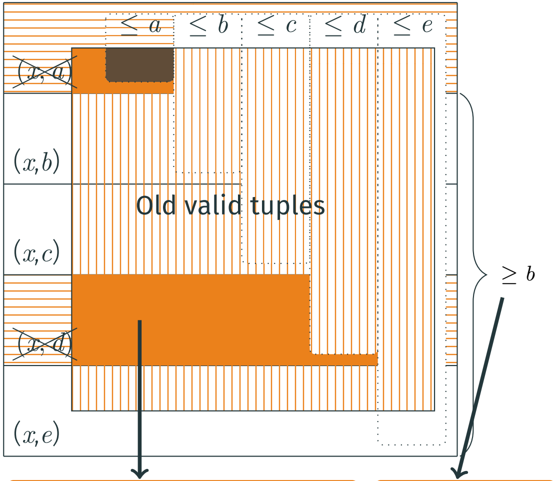
classical update



$$\texttt{currTable} \cap \left(\begin{array}{c} \texttt{support}^*[x,a] \\ \cup \\ \texttt{support}^*[x,d] \end{array} \right)^C$$

CT^{bs}: $\leq$ & $\geq$

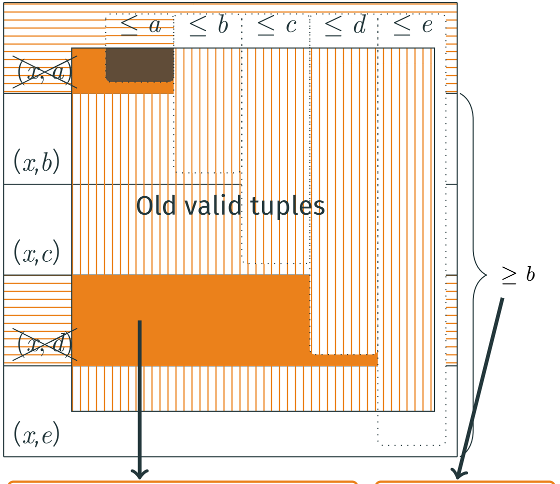
classical update



`currTable` $\cap \left(\begin{array}{c}\texttt{support}^*[x, a] \\ \cup \\ \texttt{support}^*[x, d]\end{array}\right)^C \cap$ `supportMin`$[x, b]$

CT^{bs}: $\leq$ & $\geq$

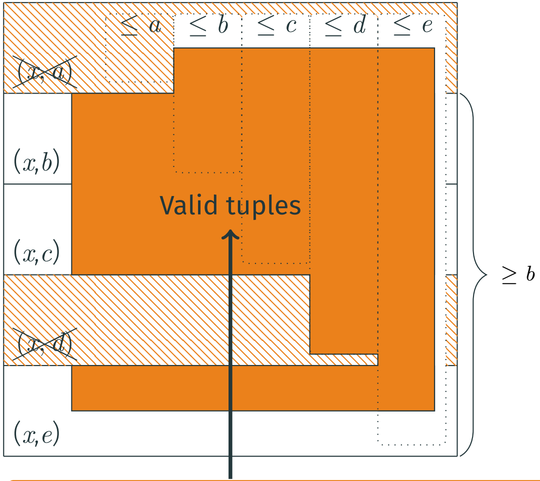
classical update



currTable $\cap$ (~~support$^*[x,a]$~~ $\cup$ support$^*[x,d]$)^C $\cap$ supportMin$[x,b]$

CT^{bs}: $\leq$ & $\geq$

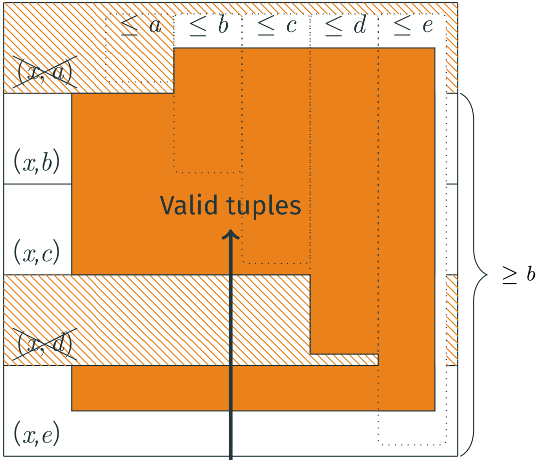
classical update



currTable $\cap \left(\begin{matrix}\cancel{\texttt{support*}[x,a]} \\ \cup \\ \texttt{support*}[x,d]\end{matrix}\right)^C \cap$ supportMin$[x,b]$

CT^{bs}: $\leq$ & $\geq$

classical update



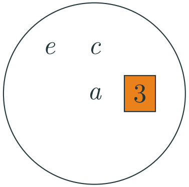
$\texttt{currTable} \cap \left(\begin{array}{c}\cancel{\texttt{support}^*[x,a]}\\ \cup \\ \texttt{support}^*[x,d]\end{array}\right)^C \cap \texttt{supportMin}[x,b]$

Algorithm: ClassicalUpdate(x)

```
1 mask ← $0$ ;
2 foreach value $a \in \Delta_x$ do
3     if $a \in [dom(x).min; dom(x).max]$
        then
4         mask ← mask |
            support*$[x, a]$ ;
5 mask ← $\sim$ mask ;
6 mask ← mask &
    supportsMin$[x, dom(x)$.min$]$ ;
7 mask ← mask &
    supportsMax$[x, dom(x)$.max$]$ ;
8 currTable ← currTable & mask ;
```

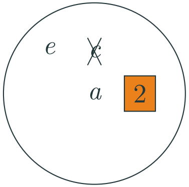
CT^{bs}: $\in$ S

update



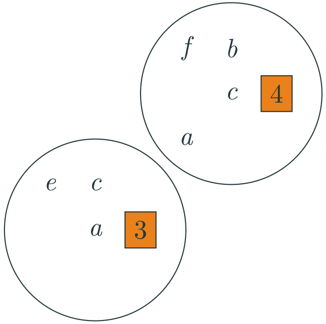
CT^{bs}: $\in$ S

update

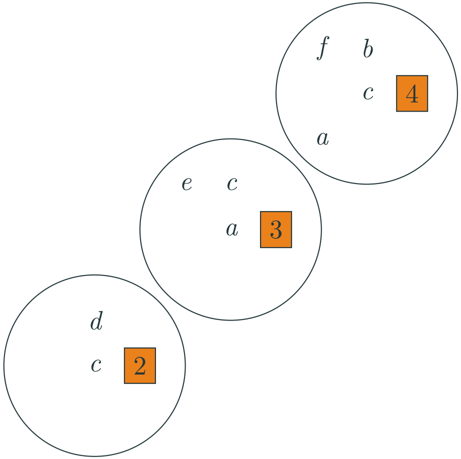


CT^{bs}: $\in$ S

update

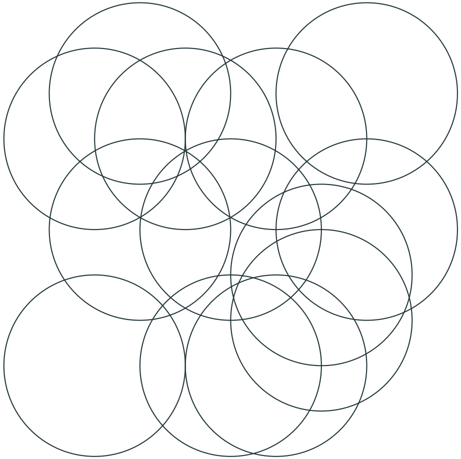


CT^{bs}: $\in$ S



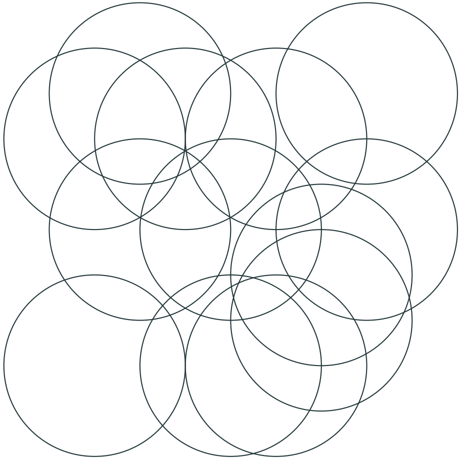
CT^{bs}: $\in$ S

update



CT^{bs}: $\in$ S

update



Algorithm: ResetUpdate(x)

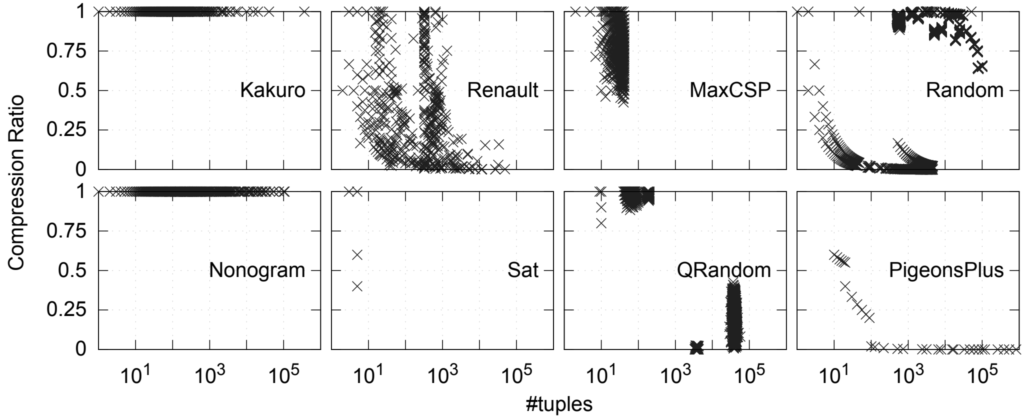
```
1 mask $\leftarrow 0$ ;
2 foreach value $a \in dom(x)$ do
3     mask $\leftarrow$ mask | supports$[x, a]$ ;
4 currTable $\leftarrow$ currTable & mask ;
```

CT^{bs}

```
Algorithm: Update(x)
1 foreach variable $x \in \texttt{scp}_{no \in S}$ do
2     if $|\Delta_x| < |dom(x)|$ then
3         ClassicalUpdate(x);
4     else
5         ResetUpdate(x);
6 foreach variable $x \in \texttt{scp}_{with \in S}$ do
7     ResetUpdate(x);
```

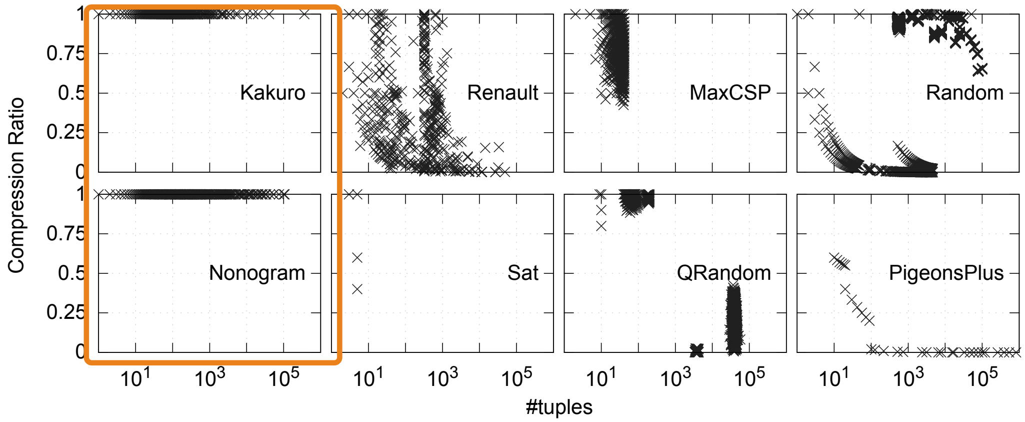
RESULTS

COMPRESSION ALGORITHM



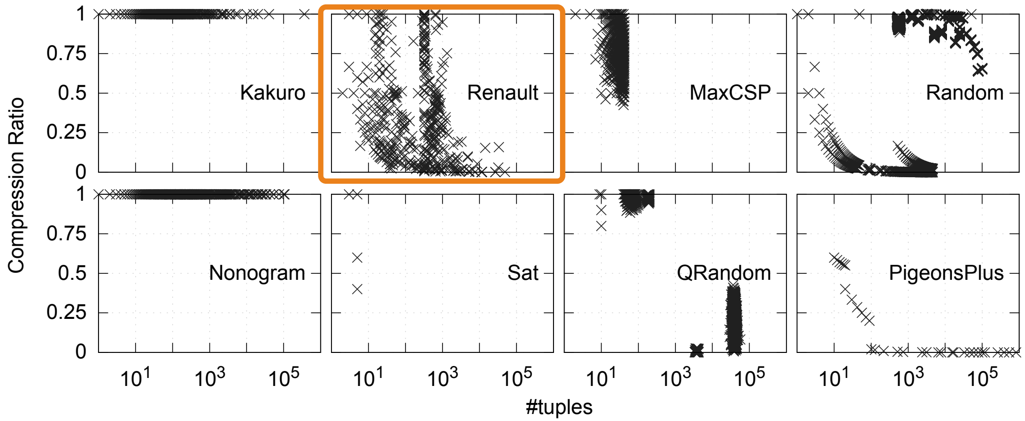
Greedy compression algorithm generating $\leq$ and $\geq$

COMPRESSION ALGORITHM



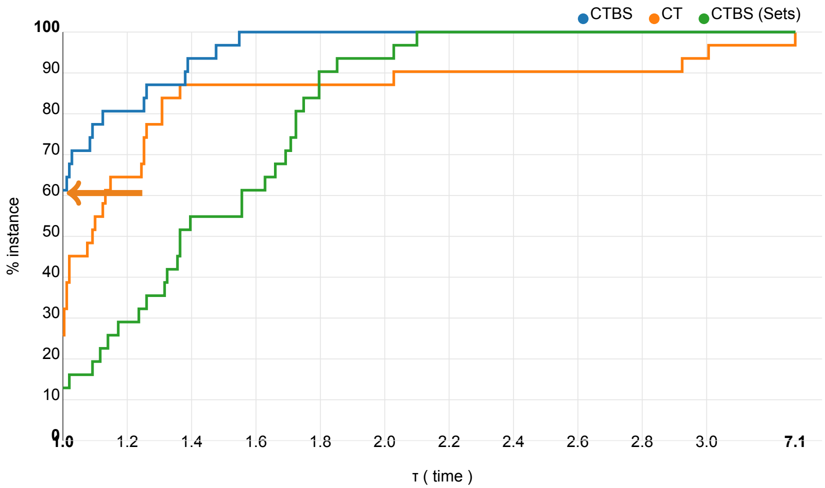
Greedy compression algorithm generating $\leq$ and $\geq$

COMPRESSION ALGORITHM



Greedy compression algorithm generating $\leq$ and $\geq$

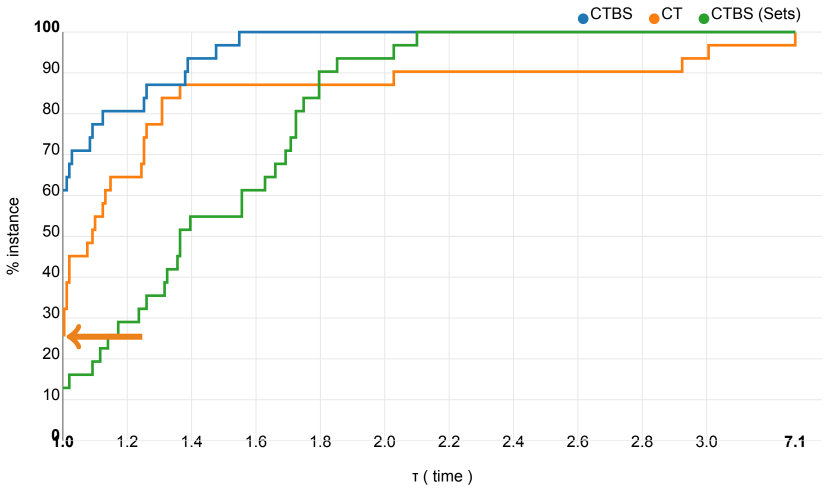
RESULTS



CTBS: compression $\leq$ & $\geq$ + post processing $\neq$ & $*$

CTBS (Sets): same compressions but seen as $\in S$

RESULTS



CTBS: compression $\leq$ & $\geq$ + post processing $\neq$ & $*$

CTBS (Sets): same compressions but seen as $\in S$

REMARK ON $\in$ S

$\|dom(x)\|$	sets		structured sets	
1	$\{a\}$	1	$*$	1
2	$\{a\}, \{b\}, \{a, b\}$	3	$a, b, *$	3
3	$\{a\}, \{b\}, \{c\}, \{a, b\},$ $\{a, c\}, \{b, c\}, \{a, b, c\}$	7	$a, b, c, \neq a,$ $\neq b, \neq c, *$	7
4	$\{a\}, \{b\}, \ldots, \{a, b\},$ {a,c}, {a,d}, {b,c}, {b,d}, $\{c, d\},$ $\{a, b, c\}, \ldots, \{a, b, c, d\}$	15	$a, b, c, d,$ $\leq b, \geq c, \neq a,$ $\neq b, \neq c, \neq d, *$	11
5	$\{a\}, \{b\}, \ldots, \{a, b\},$ {a,c}, {a,d}, {a,e}, {b,c}, {b,d}, {b,e}, {c,d}, {c,e}, $\{a, b, c\},$ {a,b,d}, {a,b,e}, {a,c,d}, {a,c,e}, $\ldots, \{a, b, c, d\}, \ldots, \{a, b, c, d, e\}$	31	a, b, c, d, e $\leq b, \leq c, \geq c,$ $\geq d, \neq a, \neq b,$ $\neq c, \neq d, \neq e, *$	15

CONCLUSION

CONCLUSION

$$\begin{tikzcd}\text{CT} \arrow[r] & \text{CT}^* \arrow[r] & \text{CT}_{bs}\end{tikzcd}$$

Tuples

CT

[CP2016]

	x	y	z
τ_1	a	a	b
τ_2	b	c	a
τ_3	b	a	a
τ_4	c	b	c

Short tuples

CT*

[AAAI17]

	x	y	z
τ_1	$*$	$*$	b
τ_2	b	c	a
τ_3	b	$*$	$*$
τ_4	c	b	$*$

Basic Smart Tuples

CT_{bs}

[CP2017]

	x	y	z
τ_1	$*$	$*$	$\in \{a, b\}$
τ_2	$\neq a$	c	$\leq a$
τ_3	b	$*$	$*$
τ_4	$\geq c$	$\neq b$	a

CONCLUSION

Tuples | Short tuples | Basic Smart Tuples

$$\begin{tikzcd}\mathrm{CT} \arrow[r] & \mathrm{CT}^* \arrow[r] & \mathrm{CT}_{bs}\end{tikzcd}$$

[CP2016] | [AAAI17] | [CP2017]

	x	y	z
τ_1	a	a	b
τ_2	b	c	a
τ_3	b	a	a
τ_4	c	b	c

	x	y	z
τ_1	$*$	$*$	b
τ_2	b	c	a
τ_3	b	$*$	$*$
τ_4	c	b	$*$

	x	y	z
τ_1	$*$	$*$	$\in \{a, b\}$
τ_2	$\neq a$	c	$\leq a$
τ_3	b	$*$	$*$
τ_4	$\geq c$	$\neq b$	a

- Increase expressiveness

CONCLUSION

Tuples | Short tuples | Basic Smart Tuples

$$\begin{tikzcd} \text{CT} \arrow[r] & \text{CT}^* \arrow[r] & \text{CT}_{bs} \end{tikzcd}$$

[CP2016] | [AAAI17] | [CP2017]

	x	y	z
τ_1	a	a	b
τ_2	b	c	a
τ_3	b	a	a
τ_4	c	b	c

	x	y	z
τ_1	$*$	$*$	b
τ_2	b	c	a
τ_3	b	$*$	$*$
τ_4	c	b	$*$

	x	y	z
τ_1	$*$	$*$	$\in \{a, b\}$
τ_2	$\neq a$	c	$\leq a$
τ_3	b	$*$	$*$
τ_4	$\geq c$	$\neq b$	a

- Increase expressiveness
- Decrease storage memory

CONCLUSION

Tuples | Short tuples | Basic Smart Tuples

$$\begin{tikzcd} \text{CT} \arrow[r] & \text{CT}^* \arrow[r] & \text{CT}_{bs} \end{tikzcd}$$

[CP2016] | [AAAI17] | [CP2017]

	x	y	z
τ_1	a	a	b
τ_2	b	c	a
τ_3	b	a	a
τ_4	c	b	c

	x	y	z
τ_1	$*$	$*$	b
τ_2	b	c	a
τ_3	b	$*$	$*$
τ_4	c	b	$*$

	x	y	z
τ_1	$*$	$*$	$\in \{a, b\}$
τ_2	$\neq a$	c	$\leq a$
τ_3	b	$*$	$*$
τ_4	$\geq c$	$\neq b$	a

- Increase expressiveness
- Decrease storage memory

- Increase speed

CONCLUSION

Tuples | Short tuples | Basic Smart Tuples

$$\begin{tikzcd} \mathrm{CT} \arrow[r] & \mathrm{CT}^* \arrow[r] & \mathrm{CT}_{bs} \end{tikzcd}$$

[CP2016] | [AAAI17] | [CP2017]

	x	y	z
τ_1	a	a	b
τ_2	b	c	a
τ_3	b	a	a
τ_4	c	b	c

	x	y	z
τ_1	$*$	$*$	b
τ_2	b	c	a
τ_3	b	$*$	$*$
τ_4	c	b	$*$

	x	y	z
τ_1	$*$	$*$	$\in \{a, b\}$
τ_2	$\neq a$	c	$\leq a$
τ_3	b	$*$	$*$
τ_4	$\geq c$	$\neq b$	a

- Increase expressiveness
- Decrease storage memory

- Increase speed
- Increase efficiency

Thank you for listening!

Any questions?